Statistics Advanced - 2| Assignment

Question 1: What is hypothesis testing in statistics?

Answer: Hypothesis testing is a formal statistical procedure for deciding whether there is enough evidence in a sample of data to support a particular claim (hypothesis) about a population parameter. It starts with a null hypothesis (H_0), defines an alternative hypothesis (H_1 or H_a), selects a test statistic and significance level (α), computes the statistic from the sample, and uses its sampling distribution to produce a p-value or compare to critical values. Based on that, we either reject H_0 or fail to reject H_0, drawing conclusions with a controlled Type I error probability.

Question 2: What is the null hypothesis, and how does it differ from the alternative hypothesis?

Answer: Null hypothesis (H_0): The default position — typically a statement of no effect or no difference (e.g., $\mu = \mu_0$).

Alternative hypothesis (H_1 or H_a): The statement you want evidence for (e.g., $\mu \neq \mu_0$, $\mu > \mu_0$, or $\mu < \mu_0$).
They are complementary. Hypothesis testing evaluates whether observed data are sufficiently incompatible with H_0 to support H_1, at the chosen significance level.

Question 3: Explain the significance level in hypothesis testing and its role in deciding the outcome of a test.

Answer: The significance level α is the threshold probability of making a Type I error (rejecting H_0 when it is actually true). Common α values are 0.05 or 0.01. If the p-value $\leq \alpha$, we reject H_0; otherwise, we fail to reject H_0. α therefore controls how strict we are about claiming a statistically significant result.

Question 4: What are Type I and Type II errors? Give examples of each.

Answer: Type I error (α): Rejecting H_0 when H_0 is true. Example: concluding a new drug works when it actually does not.

Type II error (β): Failing to reject H_0 when H_1 is true. Example: concluding the new drug does not work when it actually does.
Power = $1 - \beta$ is the probability of correctly rejecting a false H_0.

Question 5: What is the difference between a Z-test and a T-test? Explain when to use each.

Answer: Z-test: Used when the sampling distribution of the test statistic is normal and the population standard deviation (σ) is known — or when the sample size is large and we approximate σ with the sample standard deviation. The test statistic uses the **normal**

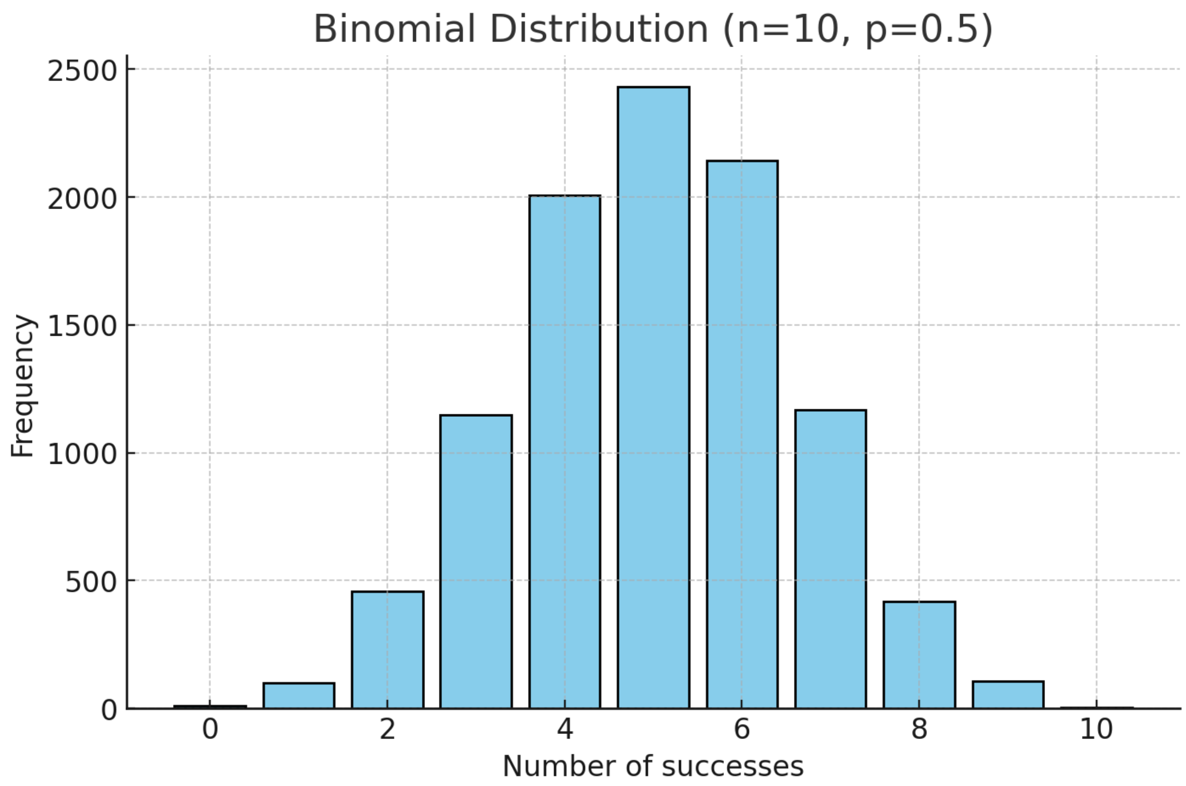
distribution.

T-test: Used when σ is unknown and the sample comes from a normally distributed population (especially for small samples). The test statistic follows Student's t-distribution with $n-1$ degrees of freedom.
In practice: use a t-test when σ is unknown (most real situations); use a Z-test if σ is known or for very large samples where normal approximation is valid.

Question 6: Write a Python program to generate a binomial distribution with n=10 and p=0.5, then plot its histogram.

Answer:

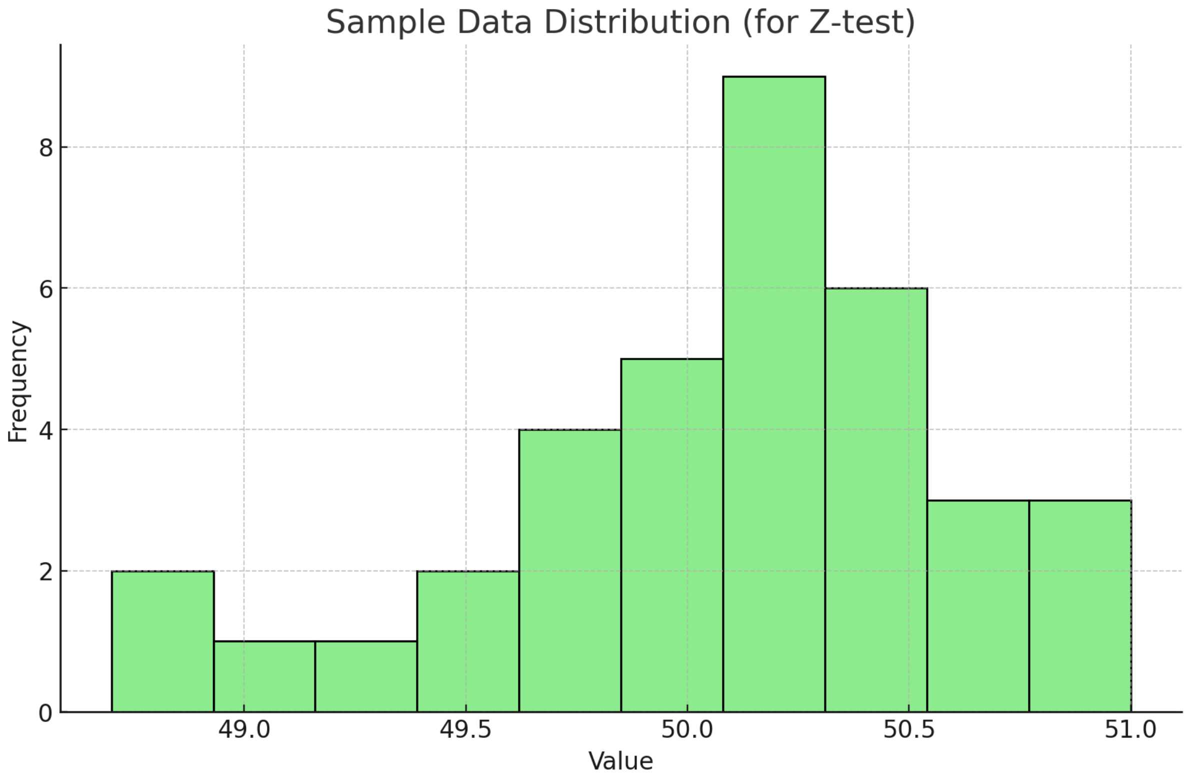
```
import numpy as np
import matplotlib.pyplot as plt

# Parameters
n = 10       # number of trials
p = 0.5      # probability of success
trials = 10000   # number of experiments

# Generate binomial distribution
binom_samples = np.random.binomial(n=n, p=p, size=trials)

# Print summary statistics
print("Mean of samples:", binom_samples.mean())
print("Standard deviation of samples:", binom_samples.std(ddof=1))

# Plot histogram
plt.figure(figsize=(8,5))
plt.hist(binom_samples, bins=range(n+2), align='left', rwidth=0.8, color="skyblue",
edgecolor="black")
plt.title("Binomial Distribution (n=10, p=0.5)")
plt.xlabel("Number of successes")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()
```



Question 7: Implement hypothesis testing using Z-statistics for a sample dataset in Python. Show the Python code and interpret the results. sample_data = [49.1, 50.2, 51.0, 48.7, 50.5, 49.8, 50.3, 50.7, 50.2, 49.6, 50.1, 49.9, 50.8, 50.4, 48.9, 50.6, 50.0, 49.7, 50.2, 49.5, 50.1, 50.3, 50.4, 50.5, 50.0, 50.7, 49.3, 49.8, 50.2, 50.9, 50.3, 50.4, 50.0, 49.7, 50.5, 49.9]

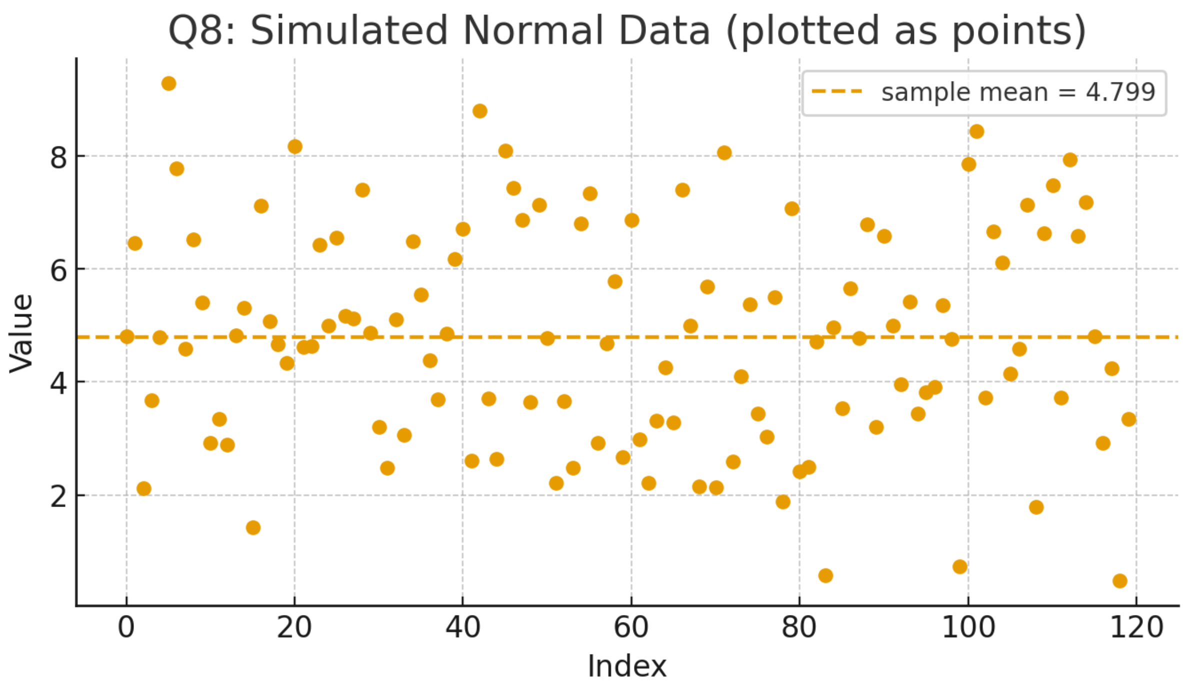
Answer:
```
import numpy as np
from math import sqrt
from scipy import stats
import matplotlib.pyplot as plt

# Sample data
sample_data = [49.1, 50.2, 51.0, 48.7, 50.5, 49.8, 50.3, 50.7, 50.2, 49.6,
               50.1, 49.9, 50.8, 50.4, 48.9, 50.6, 50.0, 49.7, 50.2, 49.5,
               50.1, 50.3, 50.4, 50.5, 50.0, 50.7, 49.3, 49.8, 50.2, 50.9,
               50.3, 50.4, 50.0, 49.7, 50.5, 49.9]

x = np.array(sample_data)
n = len(x)
sample_mean = x.mean()
sample_std = x.std(ddof=1)

# Hypothesized population mean
mu0 = 50

# Assume population sigma is known = 1.0 (for Z-test)
sigma = 1.0
```

```
# Z-test statistic
z_stat = (sample_mean - mu0) / (sigma / sqrt(n))

# Two-tailed p-value
p_val = 2 * (1 - stats.norm.cdf(abs(z_stat)))

print("Sample size:", n)
print("Sample mean:", sample_mean)
print("Sample std dev:", sample_std)
print("Z-statistic:", z_stat)
print("p-value:", p_val)

# Interpretation
alpha = 0.05
if p_val < alpha:
    print("Reject H0: Evidence suggests the mean is different from 50.")
else:
    print("Fail to reject H0: No strong evidence mean differs from 50.")

# Plot histogram of sample data
plt.hist(x, bins=10, color="lightgreen", edgecolor="black")
plt.title("Sample Data Distribution (for Z-test)")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()
```



Question 8: Write a Python script to simulate data from a normal distribution and calculate the 95% confidence interval for its mean. Plot the data using Matplotlib.

Answer:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from math import sqrt

# Parameters for the normal distribution
np.random.seed(42)       # for reproducibility
mu_true = 5.0            # true mean
sigma_true = 2.0         # true standard deviation
n = 120                  # sample size

# Simulate data
data = np.random.normal(loc=mu_true, scale=sigma_true, size=n)

# Sample statistics
sample_mean = data.mean()
sample_std = data.std(ddof=1)
se = sample_std / sqrt(n)

# 95% confidence interval using t-distribution
alpha = 0.05
t_crit = stats.t.ppf(1 - alpha/2, df=n-1)
ci_lower = sample_mean - t_crit * se
ci_upper = sample_mean + t_crit * se

print("Sample mean:", sample_mean)
print("Sample standard deviation:", sample_std)
print("95% Confidence Interval: [{:.4f}, {:.4f}]".format(ci_lower, ci_upper))

# Plot the data
plt.figure(figsize=(8,5))
plt.plot(data, marker='o', linestyle='none', alpha=0.7, color="purple")
plt.axhline(sample_mean, color="red", linestyle="--", label=f"Sample Mean =
{sample_mean:.2f}")
plt.axhline(ci_lower, color="blue", linestyle="--", label=f"95% CI Lower = {ci_lower:.2f}")
plt.axhline(ci_upper, color="blue", linestyle="--", label=f"95% CI Upper = {ci_upper:.2f}")
plt.title("Simulated Normal Data with 95% Confidence Interval")
plt.xlabel("Observation Index")
plt.ylabel("Value")
plt.legend()
plt.grid(True)
plt.show()
```



Question 9: Write a Python function to calculate the Z-scores from a dataset and visualize the standardized data using a histogram. Explain what the Z-scores represent in terms of standard deviations from the mean.

Answer:
```
import numpy as np
import matplotlib.pyplot as plt

# Function to calculate Z-scores
def calculate_z_scores(data):
    arr = np.array(data)
    mean = arr.mean()
    std = arr.std(ddof=0)   # population standard deviation
    z_scores = (arr - mean) / std
    return z_scores

# Example dataset (you can replace with any data)
sample_data = [49.1, 50.2, 51.0, 48.7, 50.5, 49.8, 50.3, 50.7, 50.2, 49.6,
               50.1, 49.9, 50.8, 50.4, 48.9, 50.6, 50.0, 49.7, 50.2, 49.5,
               50.1, 50.3, 50.4, 50.5, 50.0, 50.7, 49.3, 49.8, 50.2, 50.9,
               50.3, 50.4, 50.0, 49.7, 50.5, 49.9]

# Calculate Z-scores
zs = calculate_z_scores(sample_data)

# Print first few Z-scores
print("First 10 Z-scores:", np.round(zs[:10], 3))
print("Mean of Z-scores:", np.round(zs.mean(), 5))
print("Std of Z-scores:", np.round(zs.std(ddof=0), 5))
```

```
# Plot histogram
plt.figure(figsize=(8,5))
plt.hist(zs, bins=10, color="orange", edgecolor="black")
plt.title("Histogram of Z-scores (Standardized Data)")
plt.xlabel("Z-score")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()
```